Fan Clouds – An Alternative To Meshes

Lars Linsen[1] and Hartmut Prautzsch[2]

[1] Center for Image Processing and Integrated Computing (CIPIC)
University of California, Davis*
llinsen@ucdavis.edu
[2] Institut für Betriebs- und Dialogsysteme (IBDS)
Universität Karlsruhe, Germany**
prau@ira.uka.de

Abstract. A fan cloud is a set of triangles that can be used to visualize and work with point clouds. It is fast to compute and can replace a triangular mesh representation: We discuss visualization, multiresolution reduction, refinement, and selective refinement. Algorithms for triangular meshes can also be applied to fan clouds. They become even simpler, because fans are not interrelated. This localness of fan clouds is one of their main advantages. No remeshing is necessary for local or adaptive refinement and reduction.

1 Introduction

The real-time rendering of complex three-dimensional scenes is a challenging problem in computer graphics. Multiresolution methods can be used to reduce the complexity of a scene by adapting the level of detail to the viewing parameters.

In this paper, we present multiresolution methods for point clouds or, more precisely, for fan clouds. Fan clouds are sets of local triangulations and obviate a more costly triangular mesh generation. In Section 3, we present their construction. The visualization of fan clouds is straight-forward.

Contrary to the point cloud rendering techniques in [1, 29, 40, 44, 47], fan clouds provide a surface representation that can also be used for surface modeling and other processing operations. Thus, for most purposes, fan clouds can substitute triangular mesh representations. However, if required, triangular meshes can be generated quickly from fan clouds as discussed in Section 4.

In Section 5, we define an entropy or importance measurement for the points of a point cloud and use it for data reduction. The entropy of a point depends on its surrounding fan and is also defined for triangular meshes. Each reduction step leads to a coarser level of detail and we get a hierarchical sequence of fan cloud representations.

* http://graphics.cs.ucdavis.edu
** http://i33www.ira.uka.de

In Section 6, we go into multiresolution for modeling purposes, where detail is stored in a local frame. In Section 7, we discuss refinement for fan clouds beyond the given resolution.

Furthermore, we look at selective refinement of fan clouds for static (Section 8) as well as dynamic scenes, where the shape of the objects may change due to modeling, animation or simulation processes (Section 6). In particular, we apply selective refinement to terrain rendering in Section 8. In contrast to triangular meshes, fan clouds allow for adaptive refinement without any topological restrictions and dependencies.

Since fan clouds are local triangulations, all the introduced fan cloud processing algorithms can be run in parallel as well as by using out-of-core techniques when dealing with large-scale objects.

2 Related work

Wavelets for multiresolution methods in computer graphics are discussed in detail by Stollnitz et al. in [45] for curves and meshes with subdivision connectivity. While Eck et al. [17] remesh arbitrary meshes to generate subdivision connectivity, Kobbelt et al. [31] as well as Guskov et al. [21] develop multiresolution for arbitrary meshes. In [28], multiresolution is further generalized to non-manifolds models. A survey is given in [32].

Mesh reduction is often based on the deviation of the reduced mesh from the original one. Such a reduction approach is common, e. g., in terrain rendering (cf. [14, 16, 20, 27, 33, 38, 43, 46]). In addition, one can take into account topological aspects [24], edge lengths [24], curvatures [22, 30], normals and colors [8], or textures [10]. In all these approaches, some energy is defined and minimized. This results in reduced meshes with high fairness.

Selective refinement is commonly based either on constrained or arbitrary mesh connectivity. Constrained mesh connectivity comes with regular height fields, in general. Duchaineau et al. [16] and Lindstrom et al. [33] use binary tree hierarchies of subdivided right-angled isosceles triangles. Gross et al. [20], Pajarola [38], and Röttger et al. [43] prefer quadtree hierarchies and subdivide a rectangle into four equal rectangles. For the visualization, each rectangle is split into two triangles.

All these approaches suffer from the same problem shown in Figure 1 for quadtrees. Not all refinement steps lead to a valid mesh. In the figure, cracks appear in the surface due to the different refinement levels of adjacent quadtree blocks. Gross et al. [20] overcome this problem by generating a look-up table of all valid refinements, while Pajarola [38] as well as Röttger et al. [43] define restricted quadtrees and preserve the restriction by extra refinement steps. Restricted quadtrees requirement means that adjacent quadtree blocks differ by at most one level in the hierarchy.

This constrained mesh connectivity requires more triangles for a given accuracy than arbitrary meshes (cf. [14, 25–27, 46]). De Floriani et al. [14] use a multi-triangulation for a multiresolution representation of surfaces. They define

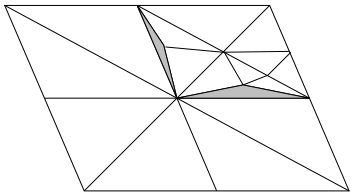


Fig. 1. Corresponding surface after selective refinement of height fields using a quadtree hierarchy.

local refinement steps and dependencies between them, which lead to a partial order of the refinement steps stored in a directed acyclic graph. Each subgraph that contains all parents of each of its nodes provides an arbitrary mesh representation of the surface.

Hoppe [25–27] as well as Xia and Varshney [46] use the vertex split refinement operation and its inverse, shown in Figure 2, to locally modify the level of detail of an arbitrary mesh representation. For a vertex split, they require that the four triangles $\Delta_1, \ldots, \Delta_4$ are active. If one of them is not active, other refinement steps are needed before.

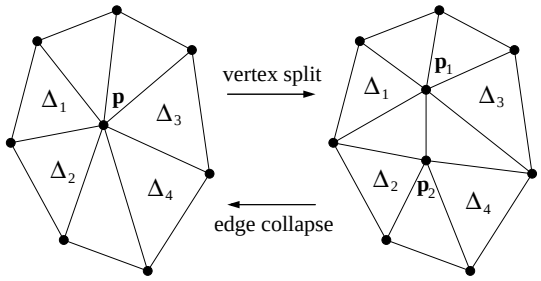


Fig. 2. A vertex split is executed only under certain preconditions.

By a sequence of edge collapses a fine mesh M^n can be transformed into a coarsest mesh M^0 [26, 27, 46]. The vertices of M^0, M^n, and all intermediate meshes M^i form a vertex hierarchy. It can be represented by a forest, in which the root nodes are the vertices of M^0 and the leaf nodes are the vertices of M^n. Any intermediate mesh M^i corresponds to a vertex front through the vertex hierarchy and represents a selective refinement of M^0. An example is illustrated by the colored nodes in Figure 3. Note that there may be invalid vertex fronts due to the restrictions for vertex splits. Establishing the forest can be done off-line in a preprocessing, which reduces the computations for real-time applications during runtime.

Point clouds can be rendered without generating a triangular mesh: One can use splatting methods [1, 29, 40, 44, 47] (also called point-based rendering), which

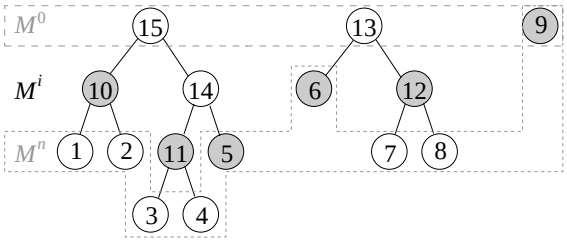


Fig. 3. A selectively refined mesh M in the vertex hierarchy, which is represented by a forest.

require dense point clouds and need much preprocessing, or fan clouds [34, 35]. In [39], point clouds are approximated by linear B-spline patches to perform modeling operations.

3 Fan Clouds

An object can be represented by a sufficiently dense set of points on its surface. Fan clouds are simple, very local structures to work with point clouds. They have been introduced and discussed in [34]. To prepare for the following, we briefly recall their construction.

For each point $\mathbf{p}$ of a point cloud, one computes a k-neighborhood consisting of k pointers to points $\mathbf{p}_1, \ldots, \mathbf{p}_k$ of the cloud close to $\mathbf{p}$ as described further below. The neighbors are determined such that the k triangles $\mathbf{p}\mathbf{p}_i\mathbf{p}_{i+1}$ form a fan that approximates the neighborhood of $\mathbf{p}$ on the surface represented by the point cloud. The set of all triangle fans is what is called a fan cloud.

To determine a k-neighborhood of a point $\mathbf{p}$, one determines the k nearest neighbors $\mathbf{p}_1, \ldots, \mathbf{p}_k$, computes the plane P with the least sum of squared distances to $\mathbf{p}, \mathbf{p}_1, \ldots, \mathbf{p}_k$, and projects all points into P. Then one sorts, i. e., permutes the indices of $\mathbf{p}_1, \ldots, \mathbf{p}_k$, such that the projections $\mathbf{q}_i$ of $\mathbf{p}_i$ lead to an increasing sequence of angles $\varphi_i = \angle \mathbf{q}_1\mathbf{q}\mathbf{q}_i$, where $\mathbf{q}$ is the projection of $\mathbf{p}$. In this order, the points $\mathbf{p}_i$ form a triangle fan or k-neighborhood of $\mathbf{p}$, see Figure 4.

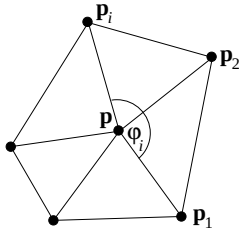


Fig. 4. A k-neighborhood for $k = 5$.

If the point density varies sharply around $\mathbf{p}$, then the neighborhood may not enclose $\mathbf{p}$. Therefore, if $\nabla\varphi_i = \varphi_i - \varphi_{i-1} > 90°$, one replaces $\mathbf{p}_k$ by the $(k+1)$st neighbor and if necessary by further next neighbors till the angle criterion $\nabla\varphi_i \leq 90°$ is met, or a certain threshold number of replacements has been reached.

Along sharp edges, the best fitting plane may be normal to the surface, see Figure 5. Therefore, if the angle criterion cannot be satisfied, we rotate the fitting plane around the axis $\mathbf{q}_{i-1}\mathbf{q}_i$ by 90° and try again to build the neighborhood.

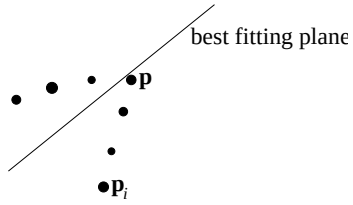


Fig. 5. Best fitting plane for an edge point viewed along the plane.

If the angle criterion can still not be met, we assume that $\mathbf{p}_{i-1}$, $\mathbf{p}$, $\mathbf{p}_i$ lie on the boundary of the surface.

Note that a triangular mesh with n vertices has about $2n$ triangles, whereas a fan cloud consists of kn triangles, where $k = 6$ is a typical number we have used. However, storage costs are not higher for fan clouds. For each point $\mathbf{p}$, we store a list of pointers to its neighbors $\mathbf{p}_1, \ldots, \mathbf{p}_k$. This is also the most efficient way to store a triangular mesh.

4 Triangular mesh generation from fan clouds

Triangular meshes are commonly used in Computer Graphics to represent surfaces. Therefore, in the nineties, various approaches were presented to generate triangular meshes from point clouds. The algorithms are based on spatial subdivision (e.g. [2, 4, 13, 23]), Delaunay tetrahedrization (e.g. [3, 7, 18]), distance functions (e.g. [13, 23]), warping (e.g. [2]), and incremental surface-increase (e.g. [5, 7, 12, 19, 36]). A survey is given in [37].

From their construction, we cannot expect fan clouds to provide a continuous surface representation. However, in all our experiments, we found that fan clouds are very much like triangular meshes, see Figure 6(a). In fact, they contain the triangles of a connected triangular mesh as a subset. Many triangles in our fan cloud are identical. Without duplicates, the fan clouds have about $2.5n$ triangles. Further, there are quadrilateral regions covered by three or four triangles of a fan cloud, i.e., by one or two superfluous triangles. Removing these superfluous triangles reduces the number of triangles to about $2.1n$, see Figure 6(b).

We observed that the reduced fan clouds are already triangular meshes with regions that are covered by several different triangulations. To obtain a 2D mesh

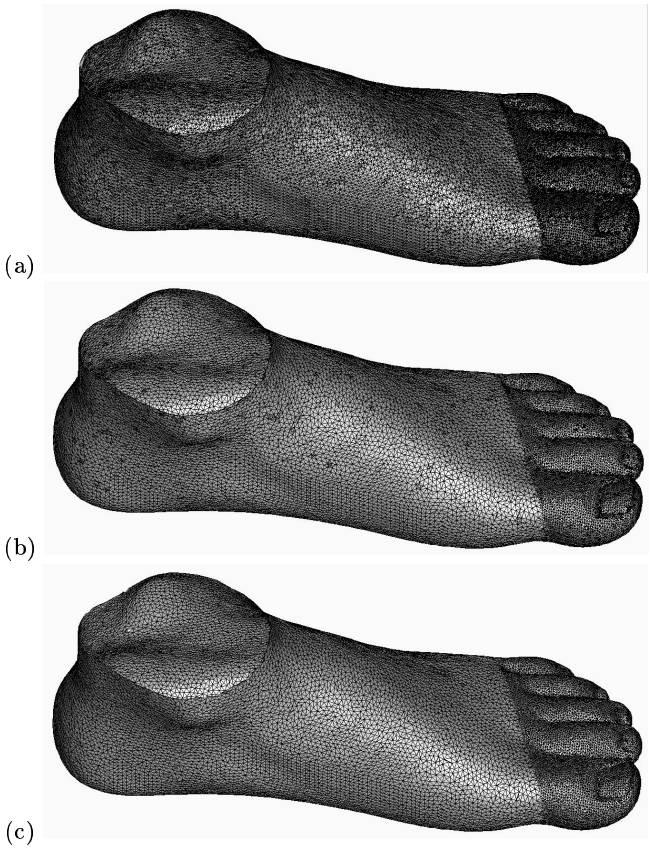


Fig. 6. The fan cloud in (a) is reduced to $2.1n$ triangles in (b) and further reduced to a triangular mesh in (c).

manifold from a reduced fan cloud, we simply grew a triangular mesh by successively adding on triangles. The result is illustrated in Figure 6(c). It has few (0.01%) self-overlaps, since we neglected any geometric aspects and based the construction only on topological aspects. Since by construction there are no edges with three ore more coincident triangles, the overlaps correspond to holes that fold back onto themselves as illustrated in Figure 7, where we have a quadrilateral hole marked by heavy lines. It is possible to avoid self-overlaps and to prove, then, the correctness of the triangulation [41].

Shaded versions of the objects in Figures 6(a) and 6(c) are shown in Figures 8(a) and 8(b), respectively.

This triangulation method is fast. The construction is similar to the approach given by [12], but uses no additional information such as point classification. The computation times of triangular mesh generations via fan clouds (incl. fan cloud generation) are given in Table 1. Since the neighborhoods for typical point clouds

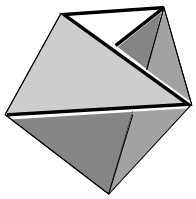


Fig. 7. Self-overlapping four sided hole (heavy lines).

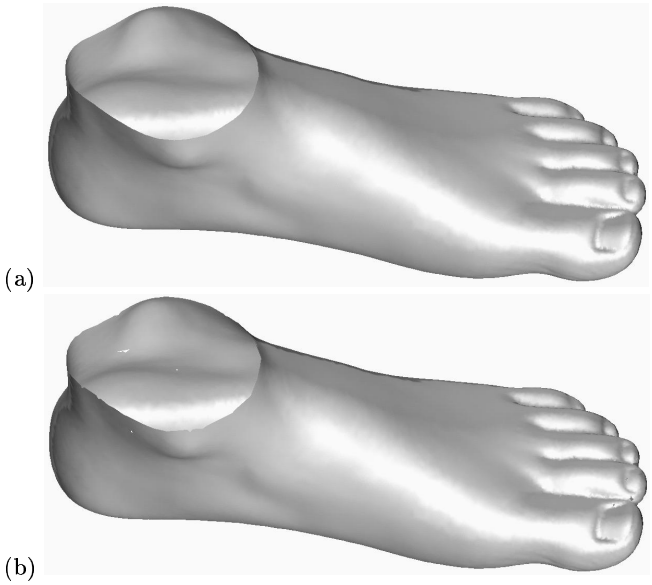


Fig. 8. Comparing fan cloud visualization (a) with triangular mesh visualization (b).

can be computed in linear time using spatial subdivision, fan clouds can be generated in linear time. For spatial subdivision, we use a 3D-cell rasterization, such that for each neighborhood estimation we only have to search in a constant number of cells containing a constant number of points.

5 Reduction

Mesh reduction means to approximate a given mesh by a coarser mesh within some tolerance. Often the underlying surface is assumed to be fair and reduction is guided by the notion that it helps to minimize some fairness energy [30]. With point clouds the reduction principle is simpler. We only remove points without constructing some approximating surface. Hence, we try to find a small subset of a given point cloud that still is a good representation for the object.

Since fan cloud generation is fast, the reduction scheme can be applied to large-scale data sets. Exploiting the local data structure of fan clouds, the reduc-

method	#points	computer	time
Fan cloud generation	47109	SGI Indigo2 Extreme	45 s
		Sun Ultra30	13 s
		PC with Athlon K7 800MHz	5 s
	100001		17 s
	160940		25 s
Global triangulation via fan cloud	20021	PC with Athlon K7 800MHz	4 s
	35948		8 s
	160940		41 s

Table 1. Computation times for fan cloud generation and global triangulation via fan cloud.

tion scheme can also make use of out-of-core techniques and distributed computing.

For fan cloud reduction, we introduce an entropy that encodes for each point, how much information it contributes to the geometric information of the surface. Similar to common fairing energies the entropy is based on point distances, color information, curvature estimates, and change in curvature. The main difference is that we try to keep the entropy high rather than to minimize it.

An object like a sphere has an homogenous shape characteristic. In such situations a point that is close to its neighbors is less important than a point with large distances $d_j := \|\mathbf{q}_j - \mathbf{p}\|_2$ to its neighbors $\mathbf{q}_1, \ldots, \mathbf{q}_k$. Thus, we define the measure

$$M_{\text{dist}}(\mathbf{p}) := \frac{1}{k} \sum_{j=1}^{k} d_j$$

for each point $\mathbf{p}$ of the point or fan cloud.

Further, we estimate the curvature at $\mathbf{p}$ in the direction of any $\mathbf{q}_j$ by $\frac{\|\mathbf{n}_j - \mathbf{n}\|_2}{d_j}$, where $\mathbf{n}$ and $\mathbf{n}_j$ are the normals at $\mathbf{p}$ and $\mathbf{q}_j$, respectively. Note that, actually, this is a (in d_j) linear approximation of $\sqrt{\kappa^2 + \tau^2} = \|\mathbf{n}'\|_2$, where κ is the normal curvature and τ the geodesic torsion at $\mathbf{p}$ in the direction $\mathbf{q}_j - \mathbf{p}$, see e. g. [6]. Hence, this estimate also measures the torsion or non-planarity of geodesics on the surface. Averaging these terms for all neighbors of $\mathbf{p}$ leads to the measure

$$M_{\text{curv}}(\mathbf{p}) := \frac{1}{k} \sum_{j=1}^{k} \frac{\|\mathbf{n}_j - \mathbf{n}\|_2}{d_j}$$

of the curvature and geodesics torsion at $\mathbf{p}$.

Defining the change of curvature at $\mathbf{p}$ by

$$M_{\text{cc}}(\mathbf{p}) := \frac{1}{k} \sum_{j=1}^{k} \frac{|M_{\text{curv}}(\mathbf{q}_j) - M_{\text{curv}}(\mathbf{p})|}{d_j}$$

and the change of color at $\mathbf{p}$ by

$$M_{\text{col}}(\mathbf{p}) := \frac{1}{k} \sum_{j=1}^{k} \frac{\|\mathbf{c}_j - \mathbf{c}\|_2}{d_j} \ ,$$

where $\mathbf{c}$ and $\mathbf{c}_j$ contain the RGB-encoded color information at $\mathbf{p}$ and $\mathbf{q}_j$, we get the entropy $M(\mathbf{p})$ at $\mathbf{p}$,

$$M(\mathbf{p}) := \alpha_1 M_{\text{dist}}(\mathbf{p}) + \alpha_2 M_{\text{curv}}(\mathbf{p}) + \alpha_3 M_{\text{cc}}(\mathbf{p}) + \alpha_4 M_{\text{col}}(\mathbf{p}) \ ,$$

with adjustable weights $\alpha_1, \ldots, \alpha_4$.

To make the terms of the entropy scale-invariant, we define the average distance between neighbored points

$$d := \frac{1}{k \cdot n} \sum_{\mathbf{p} \in P} \sum_{j=1}^{k} d_j \ ,$$

where P is the point cloud and n its size, and divide the term $M_{\text{dist}}(\mathbf{p})$ by d, multiply the terms $M_{\text{curv}}(\mathbf{p})$ and $M_{\text{col}}(\mathbf{p})$ by d and multiply the term $M_{\text{cc}}(\mathbf{p}_i)$ by d^2.

The fan cloud rendered in Figure 9(a) is reduced to 42% using the above entropy with $\alpha_1, \alpha_2, \alpha_3, \alpha_4 = 1, 0, 0, 0$ in 9(b) and $\alpha_1, \alpha_2, \alpha_3, \alpha_4 = 1, 1, 50, 0$ in 9(c). We iteratively remove a point $\mathbf{p}$ with lowest entropy and recompute the entropy of all points that had $\mathbf{p}$ as a neighbor. For Figures 9(d), 9(e), and 9(f), we zoom into one region of Figures 9(a), 9(b), and 9(c). Clearly the loss of surface features is dramatic in 9(e) and negligible in 9(f).

Figure 10 gives a comparison of reductions applied to a colored fan cloud. The fan cloud in (a) shows part of the Grand Canyon. In (b), (c), and (d), it is reduced to 17.6%. In (b), the approximation error is minimized (see Section 8). In (c) and (d), we preserved the entropy as much as possible, where $\alpha_1, \alpha_2, \alpha_3, \alpha_4 = 1, 0, 0, 0$ in (c), which results in an almost equal distribution of the points, and $\alpha_1, \alpha_2, \alpha_3, \alpha_4 = 1, 1, 50, 1$ in (d). Again, preserving the entropy maintains salient curvature and color features best.

Note that the entropy-based reduction can also be applied to triangular meshes. Furthermore, recently developed point-based rendering techniques [1, 29, 40, 44, 47] as well as hybrid (point-based/triangle-based) rendering approaches [9, 11] can be combined with this fan cloud reduction scheme.

6 Level of Detail

Successively removing points from the original fan cloud and adjusting the triangle fans, that contained the removed point, leads to a hierarchical sequence of fan clouds. We can switch back to any finer resolution in this hierarchy by reinserting points that have been removed. To accommodate for shape modifications at some lower resolution, we use for each point that we remove local rather than global object coordinates as described in [21, 31] for triangular meshes.

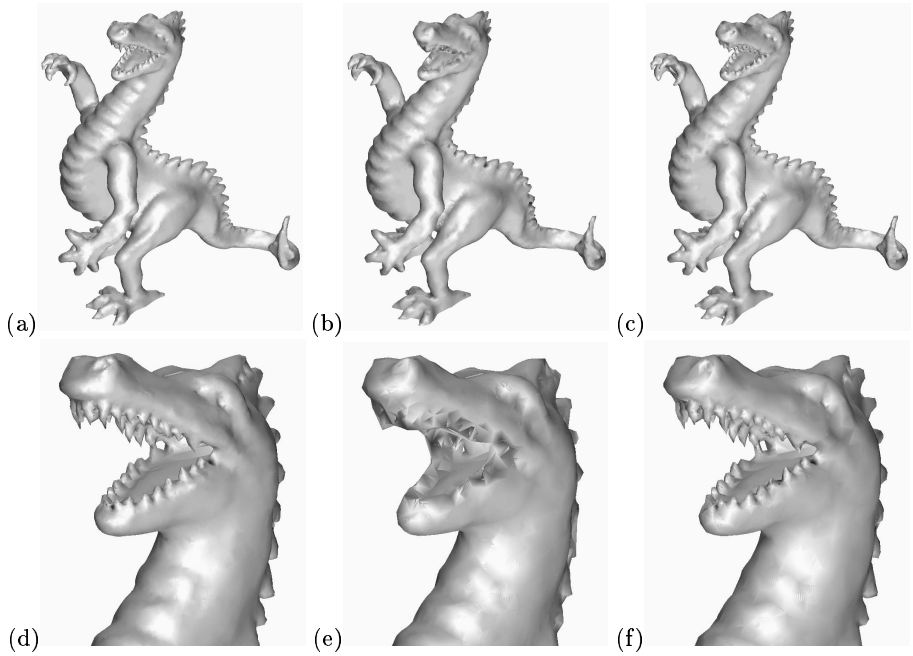


Fig. 9. Reducing a fan cloud (a),(d) to 42% only by considering distances (b),(e) or by considering distances and surface features (c),(f).

Examples and applications of the level-of-detail control combined with various modeling operations are given in [35].

Furthermore, to reduce complexity of a large object, one can speed up the visualization by a selective refinement, i. e., we extract and visualize the point cloud with a fine resolution only in the region of interest. We proceed as above, i. e., we update the positions of all points in the reverse order of their removal, but we only activate, i. e., reinsert the points within the region of interest.

In Figure 11, we give an example. To distinctly illustrate the selective refinement, the points are visualized, too. The point cloud in (a) consists of 14379 points and is reduced to 2156 points as shown in (b). In (c), we see the point cloud after a global smoothing. Finally, the face of the represented bunny is selectively refined. The result in (d) consists of only 3031 points, but in the region of interest full detail information is available for further modeling purposes.

7 Refinement

For some applications in modeling and animation, it can be necessary to refine a point cloud beyond its given resolution, or in other words to extend the hierarchical sequence above beyond its finest level.

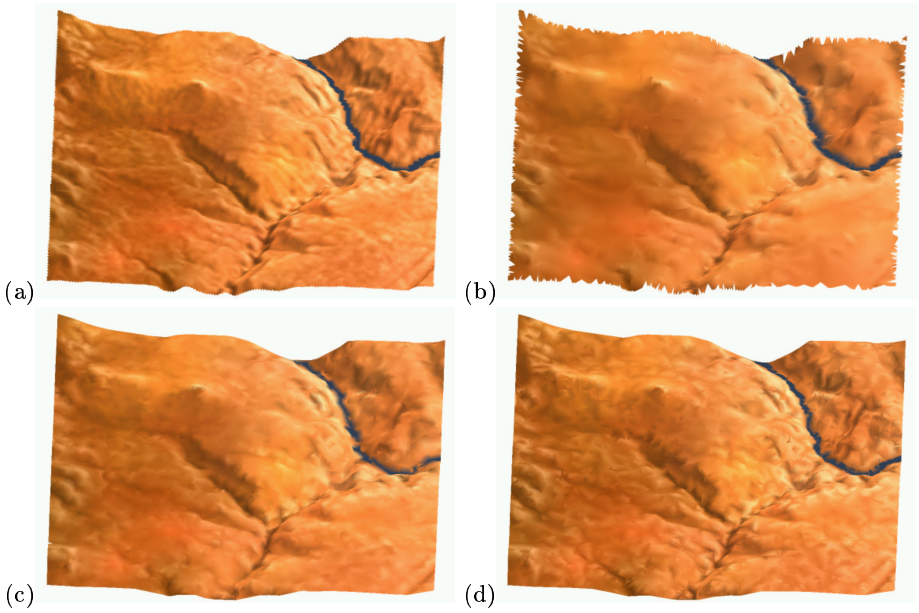


Fig. 10. Fan cloud (a) reduction based on approximation error (b), distances (c), and entropy (d).

To insert a new point $\mathbf{r}$, we determine the point $\mathbf{p}$ with highest entropy $M(\mathbf{p})$, its neighbor, say $\mathbf{q}_1$, with highest entropy, and the predecessor or successor of $\mathbf{q}_1$, say $\mathbf{q}_2$, in the neighborhood of $\mathbf{p}$ with highest entropy. We compute the weights

$$\begin{aligned}\omega &:= \frac{M(\mathbf{p})}{M(\mathbf{p})+M(\mathbf{q}_1)+M(\mathbf{q}_2)}\\ \omega_1 &:= \frac{M(\mathbf{q}_1)}{M(\mathbf{p})+M(\mathbf{q}_1)+M(\mathbf{q}_2)}\\ \omega_2 &:= \frac{M(\mathbf{q}_2)}{M(\mathbf{p})+M(\mathbf{q}_1)+M(\mathbf{q}_2)}\end{aligned}$$

and define the initial position of the new point by

$$\mathbf{r} := \omega\mathbf{p} + \omega_1\mathbf{q}_1 + \omega_2\mathbf{q}_2 \ .$$

In general, $\mathbf{r}$ does not lie on the surface represented by the point cloud. Therefore, we move $\mathbf{r}$ in the direction of the surface normal. In [15], the mean curvature normal vector is approximated by

$$\mathbf{n}_\mathbf{r} = \frac{1}{2A}\frac{\partial A}{\partial \mathbf{r}} \ ,$$

where A is the area of the fan of $\mathbf{r}$. One can easily show that this is an affine combination of the form

$$\mathbf{n}_\mathbf{r} = \alpha_0\mathbf{r} - \sum_i \alpha_i\mathbf{r}_i \ ,$$

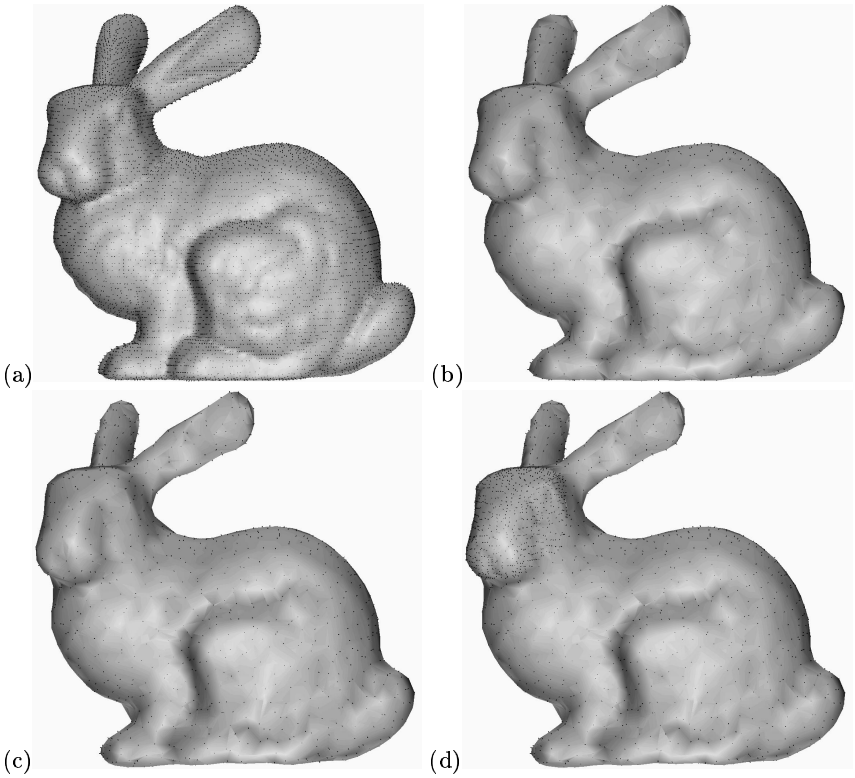


Fig. 11. Selective refinement (d) of a point cloud (a) after reduction (b) and modeling (c).

where $\mathbf{r}_i$ are the neighbors of $\mathbf{r}$. We set

$$\triangledown\mathbf{r} = \mathbf{r} - \sum_i \frac{\alpha_i}{\alpha_0}\mathbf{r}_i$$

and

$$\mathbf{r} = \overline{\mathbf{r}} := \sum_i \frac{\alpha_i}{\alpha_0}\mathbf{r}_i + \rho\frac{\triangledown\mathbf{r}}{\|\triangledown\mathbf{r}\|_2} \ ,$$

where

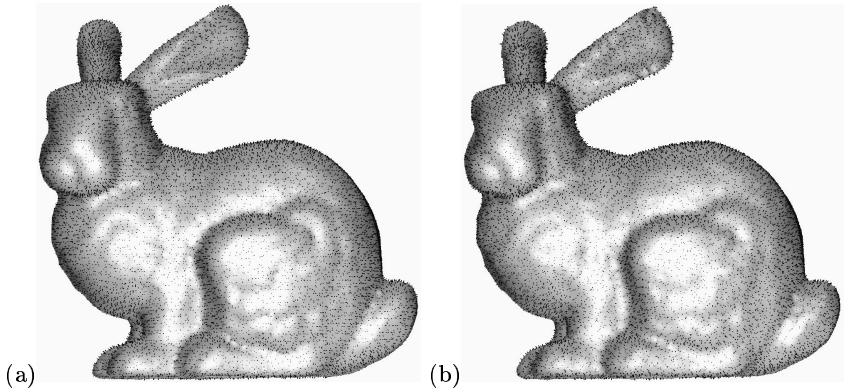
$$\rho = \omega\|\triangledown\mathbf{p}\|_2 + \omega_1\|\triangledown\mathbf{q}_1\|_2 + \omega_2\|\triangledown\mathbf{q}_2\|_2 \ .$$

Since the α_i do not change when $\mathbf{r}$ moves (not too much) in the normal direction, we get $\triangledown\overline{\mathbf{r}} \approx \rho$.

If $\mathbf{c}$, $\mathbf{c}_1$, and $\mathbf{c}_2$ are the colors of $\mathbf{p}$, $\mathbf{q}_1$ and $\mathbf{q}_2$, the new point $\mathbf{r}$ gets the color

$$\omega\mathbf{c} + \omega_1\mathbf{c}_1 + \omega_2\mathbf{c}_2 \ .$$

To validate our approach we have applied a reduction-refinement-cycle to the point cloud shown in Figure 12(a) ten times. The reduction-refinement-cycle switches between the levels 100% and 50% without storing detail information. The result in Figure 12(b) illustrates, that the shape of the object remains almost unaffected. Note that we used the definition of $\Diamond$ given in [34, 35] to produce these Figures. This definition is different but very similar to the above.



(a) (b)

Fig. 12. Fan cloud (a) and iterated reduction/refinement (b).

8 Selective Refinement

The reduction described in Section 5 is only based on local object information. However, it is also possible to base a reduction decision on global information. In particular, for large objects, it is useful to consider positions relative to the viewer and use a lower resolution for invisible or distant parts of an object.

Such a view-dependent reduction or refinement is considered in [26, 27], where Hoppe proposes a *view-dependent progressive mesh* (VDPM) framework to render large terrain models. Using fan clouds instead of meshes, we present a few improvements to Hoppe's idea and speak of a *view-dependent fan cloud* (VDFC). As described in Section 2, Hoppe [26, 27] builds (in a preprocessing step) a forest from a given mesh by successive half-edge collapses. We also build such a forest. However, rather than only using half-edge collapses, we also tried to successively merge the point $\mathbf{p}$ with lowest entropy $M(\mathbf{p})$ with its neighbor $\mathbf{q}$ with lowest entropy $M(\mathbf{q})$ into the new point

$$\mathbf{r} = \frac{M(\mathbf{p})\mathbf{p} + M(\mathbf{q})\mathbf{q}}{M(\mathbf{p}) + M(\mathbf{q})} .$$

The neighbors of $\mathbf{r}$ are the neighbors of $\mathbf{p}$ and $\mathbf{q}$ except for $\mathbf{p}$ and $\mathbf{q}$.

With this merge, we obtained better results than with the half-edge collapse, which is also used in [42, 46]. Figure 13 shows an example where we reduced the point cloud of Figure 10(a) to 17.6% by half-edge collapses (a) and by the point merges above (b).

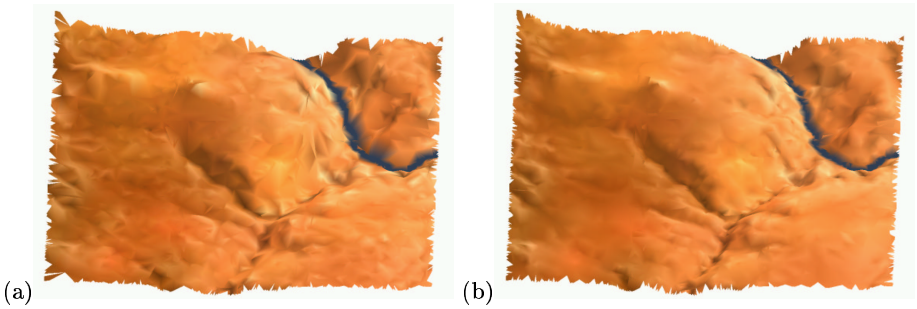


Fig. 13. Reducing the terrain of Figure 10(a) by applying half edge collapses (a) or point merges (b).

Note that, here, different points may become differently many neighbors.

As mentioned in Section 2, Hoppe allows a vertex split only if certain neighbors exist as during the collapse. Here, there are no such conditions. If, during visualization, a neighbor $\mathbf{q}$ of an active point $\mathbf{p}$ is not active, we use the active ancestor of $\mathbf{q}$ in the forest hierarchy rather than executing further refinement steps.

We do not know if fans can become invalid neighborhoods. Figure 14 depicts such a theoretical case for a triangular mesh. Hoppe does not mention this problem and we did not encounter it in all our experiments with fan clouds.

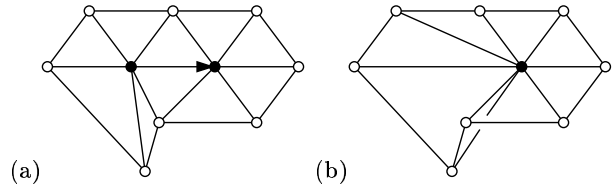


Fig. 14. Triangulation before (a) and after (b) a point merge.

The view-dependent refinement of a terrain is based on a selective refinement due to visibility, distance, and viewing direction. Points outside the view frustum are not active and the resolution increases toward the viewpoint.

To control the resolution or level of detail, we estimate for each point $\mathbf{r}$ in the forest the local deviation of its current triangle fan T from the original fan cloud. When $\mathbf{r}$ is obtained by a point merge, we compute the maximum deviation from

all its descendants to T and denote it by $e(\mathbf{r})$. Later, when we run the view-dependent visualization, $\mathbf{r}$ becomes part of the active point cloud, if it lies in the view frustum and if

$$e(\mathbf{r}) > \frac{d(\mathbf{r})}{d_{max}}\, e_{max} \ ,$$

with $d(\mathbf{r})$ the distance of $\mathbf{r}$ from the viewpoint, d_{max} the range of sight, and e_{max} the maximum approximation error.

Note that the triangle fan T used to compute $e(\mathbf{r})$ may differ from the active triangle fan of $\mathbf{r}$ used to render the object. However, we used T to be able to compute $e(\mathbf{r})$ in the preprocessing step.

Hoppe [26] as well as De Floriani et al. [14] and Xia and Varshney [46] work with triangular meshes, which they consider as piecewise linear functions over $\mathbf{R}^2$. Consequently, they define $e(\mathbf{r})$ to be the maximum functional difference between T and the relevant part of the initial fine mesh, i. e., between two approximations. Also, they compute $e(\mathbf{r})$ during the visualization.

Large terrains are usually partitioned into blocks such that adjacent blocks share the points on their common boundary. Under a so called dynamic scene management, only blocks within the range of sight are considered. For stitching blocks together, the same points on their common boundary have to be active. Thus, in the preprocessing step, we exclude boundary points from point merges. Then, after having established the forests for the single blocks, we unite the blocks and perform further point merges including the boundary points. Without these further merges, the boundaries become visible in the shaded visualization.

Figure 15 shows a view-dependent refinement of a terrain in point cloud representation. In (a), we show the point cloud and in (b) its shaded visualization. The viewing points used for the view-dependent refinement and the visualization in Figure 15 are different. In Figure 16 these viewing points are the same.

9 Conclusion

In this paper, we have shown

- that multiresolution techniques, well-known for triangular meshes, can also be applied to fan clouds,
- that entropy-based reduction of fan clouds or triangular meshes maintains characteristic features better than energy-based reduction,
- that selective refinement underlies no restrictions with fan clouds,

and we have introduced a refinement algorithm to refine a point cloud beyond its initial resolution.

Fan clouds can be computed efficiently and faster than triangular meshes. Although, in general, they do not form a 2D manifold, they can be stripped down to triangular 2D mesh manifolds.

The number of triangles in a fan cloud is higher than in a triangular mesh. However, storage costs are not higher, and, moreover, there are many duplicates,

(a)

(b)

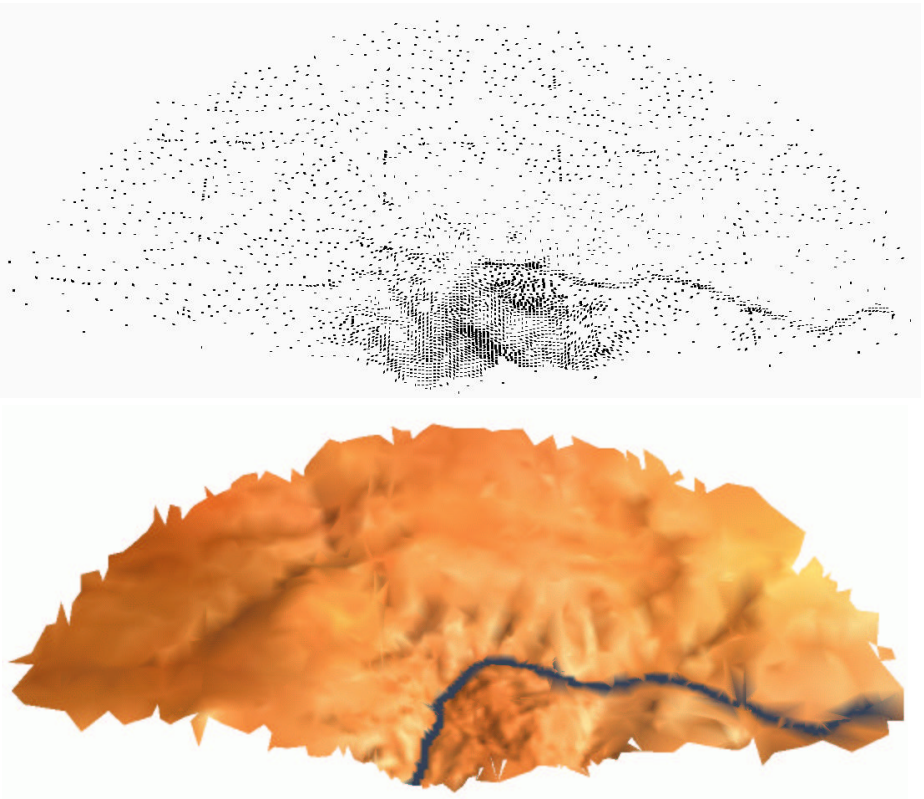


Fig. 15. View-dependent refinement of terrain using VDFC representation.

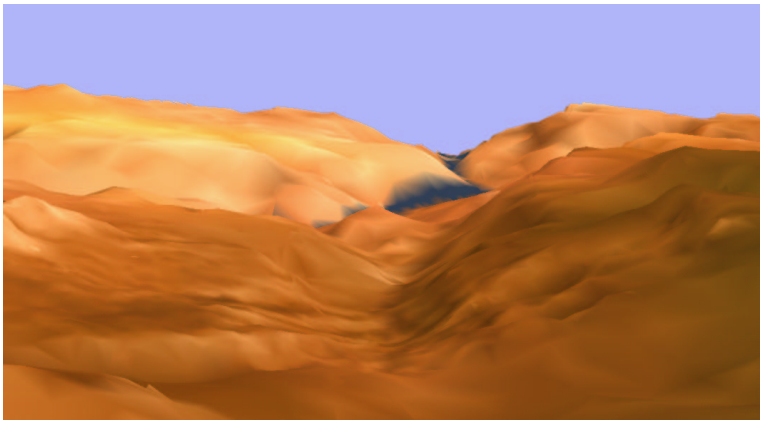


Fig. 16. A selectively refined terrain from the viewpoint.

which we do not render. In fact, in the average, we have only 2.5 different triangles per point and if we remove single redundancies even only 2.1 triangles per point. When using the VDFC framework, the average number of triangles in a selectively refined scene is even smaller, namely approximately 2.0 triangles per point when we start with $k = 6$. This is about the same as for triangular meshes.

One of the main advantages of fan clouds is their complete localness. Therefore, they are perfectly suited for local refinement, i. e., there is no remeshing necessary in the transition areas between refined and unrefined parts of a point cloud.

In particular, we can split any point at any time and need not check its neighbors as in [26]. Also different from [26], we compute the approximation error in a preprocess. Altogether, this implies that we can split points at almost no extra cost during runtime, whereas in [26] the evaluation and preparation costs for a split are four times higher than for its execution.

Moreover, since we have no restriction as to any mesh connectivity, we can reduce the number of points to its minimum given by the error tolerance.

Another advantage of the VDFC representation above is the ease, with which objects can be added to the scene. The VDFC representation of monuments, places of interest, etc. can simply be added as further separate trees to the forest of the terrain representation. Since neither the VDFC representation nor the error estimation are based on a height field over some parameter plane, these objects (and the terrain as well) can be of arbitrary shape, e. g. it is possible to represent mountain ledges. Thus, the VDFC representation provides a framework for all features of a three-dimensional geographical information system.

References

1. Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, Claudio T. Silva: *Point Set Surfaces.* Proceedings of IEEE Conference on Visualization '01, 21–28, 2001.
2. Maria-Elena Algorri, Francis Schmitt: *Surface reconstruction from unstructured 3d data.* Computer Graphics Forum, Vol. 15 (1), 47 - 60, 1996.
3. Marco Attene, Michela Spagnuolo: *Automatic surface reconstruction from point sets in space.* Computer Graphics Forum, Vol. 19 (3), 457 - 466, 2000.
4. Chandrajit Bajaj, Fausto Bernardini, Guoliang Xu: *Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans.* Proceedings of SIGGRAPH '95, 109 - 118, 1995.
5. Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Clàudio Silva, Gabriel Taubin: *The Ball-Pivoting Algorithm for Surface Reconstruction.* IEEE Transactions on Visualization and Computer Graphics, Vol. 5 (4), 349 - 359, 1999.
6. Wolfgang Boehm, Hartmut Prautzsch: *Geometric Concepts for Geometric Design.* AK Peters, Wellesley, 1994.
7. Jean-Daniel Boissonat: *Geometric Structures for Three-Dimensional Shape Representation.* ACM Transactions on Graphics, 266 - 286, 1984.
8. Swen Campagna, Hans-Peter Seidel: *Generating and Displaying Progressive Meshes.* Proceedings of 3D Image Analysis and Synthesis, Erlangen, 35–42, 1997.

9. Baoquin Chen, Minh Xuan Nguyen: *POP: A Hybrid Point and Polygon Rendering System for Large Data* Proceedings of IEEE Conference on Visualization '01, 45–52, 2001.
10. Jonathan D. Cohen, Marc Olano, Dinesh Manocha: *Appearance-Preserving Simplification.* Proceedings of SIGGRAPH '98, 115–122, 1998.
11. Jonathan D. Cohen, Daniel G. Aliaga, Weiqiang Zhang: *Hybrid Simplification: Combining Multi-resolution Polygon and Point Rendering* Proceedings of IEEE Conference on Visualization '01, 37–44, 2001.
12. Patricia Crossno, Edward Angel: *Spiraling Edge: Fast Surface Reconstruction from Partially Organized Sample Points* Proceedings of IEEE Conference on Visualization '99, 1999.
13. Brian Curless, Marc Levoy: *A Volumetric Method for Building Complex Models from Range Images.* Proceedings of SIGGRAPH '96, New Orleans, LA, 4-9 August 1996.
14. L. De Floriani, P. Magillo, E. Puppo: *VARIANT: A System for Terrain Modeling at Variable Resolution.* Geoinformatica, Vol. 4(3), 287–315, 2000.
15. Mathieu Desbrun, Mark Meyer, Peter Schröder, Alan Barr: *Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow.* Proceedings of SIGGRAPH '99, 1999.
16. M. Duchaineau, M. Wolinsky, D. Sigeti, M. Miller, C. Aldrich, M. Mineev-Weinstein: *ROAMing terrain: real-time optimally adapting meshes.* Proceedings of IEEE Visualization '97, 81–88, 1997.
17. Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, Werner Stuetzle: *Multiresolution Analysis of Arbitrary Meshes.* Proceedings of SIGGRAPH '95, 1995.
18. H. Edelsbrunner, E.P. Mücke: *Threedimensional alpha shapes.* ACM Transactions on Computer Graphics, Vol. 13 (1), 43 - 72, 1994.
19. M. Gopi, S. Krishnan, C.T. Silva: *Surface Reconstruction based on Lower Dimensional Localized Delaunay Triangulation.* Computer Graphics Forum, Vol. 19 (3), 2000.
20. M. Gross, O. Staadt, R. Gatti: *Efficient Triangular Surface Approximations using Wavelets and Quadtree Data Structures.* IEEE Transactions on Visualization and Computer Graphics, Vol. 2 (2), 130–143, 1996.
21. Igor Guskov, Wim Sweldens, Peter Schröder: *Multiresolution Signal Processing for Meshes.* Proceedings of SIGGRAPH '99, 1999.
22. Bernd Hamann: *A data reduction scheme for triangulated surfaces.* Computer Aided Geometric Design, Vol. 11, 197–214, 1994.
23. Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, Werner Stuetzle: *Surface Reconstruction from Unorganized Points.* Computer Graphics, Vol. 26, 71 - 78, 1992.
24. Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, Werner Stuetzle: *Mesh Optimization.* Computer Graphics Proceedings, Annual Conference Series, Vol. 7, 19–26, 1993.
25. Hugues Hoppe: *Progressive meshes.* Proceedings of SIGGRAPH '96, 99–108, 1996.
26. Hugues Hoppe: *View-dependent refinement of progressive meshes.* Proceedings of SIGGRAPH '97, 189–198, 1997.
27. Hugues Hoppe: *Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering.* IEEE Visualization, 35–42., 1998.
28. Andreas Hubeli, Markus Gross: *Multiresolution Methods for Non-Manifold Models.* IEEE Transaction on Visualization and Computer Graphics, 2001.

29. Aravind Kalaiah, Amitabh Varshney: *Differential Point Rendering.* Rendering Techniques '01 , S.J. Gortler and K. Myszkowski (edts.), Springer-Verlag, 139–150, 2001
30. Leif Kobbelt, Swen Campagna, Hans-Peter Seidel: *Mesh Reduction Revisited.* Universit"at Erlangen, 1997.
31. Leif Kobbelt, Swen Campagna, Jens Vorsatz, Hans-Peter Seidel: *Interactive Multi-Resolution Modeling on Arbitrary Meshes.* Proceedings of SIGGRAPH '98, 1998.
32. Leif Kobbelt: *Multiresolution techniques.* To appear in: Farin, Hoschek, Kim (Edts.), "'Handbook of Computer Aided Geometric Design"', Elsevier.
33. P. Lindstrom, D. Koller, W. Ribarsky, L. Hodges, N. Faust, G. Turner: *Real-time, continuous level of detail rendering of height fields.* Proceedings of SIGGRAPH '96, 109–118, 1996.
34. Lars Linsen, Hartmut Prautzsch: *Local Versus Global Triangulations.* Proceedings of Eurographics '01, Short Presentations, Manchester, 257–263, 2001.
35. Lars Linsen: *Oberflächenrepräsentation durch Punktwolken.* Dissertation, Universität Karlsruhe, Verlag Shaker, Aachen, 2001.
36. Robert Mencl, Heinrich Müller: *Graph-Based Surface Reconstruction Using Structures in Scattered Point Sets.* Proceedings of Computer Graphics International '98, Hannover, 1998.
37. Robert Mencl, Heinrich Müller: *Interpolation and Approximation of Surfaces from Three-Dimensional Scattered Data Points.* State of the Art Report for EUROGRAPHICS '98, Lisbon, 1998.
38. Renato Pajarola: *Large scale Terrain Visualization using the Restricted Quadtree Triangulation.* Technical Report, ETH Zürich, Switzerland, 1998.
39. Mark Pauly, Markus Gross: *Spectral Processing of Point-Sampled Geometry.* Proceedings of SIGGRAPH '01, 2001.
40. Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, Markus Gross: *Surfels: Surface Elements as Rendering Primitives.* Proceedings of SIGGRAPH '00, 2000.
41. Stephan Preuß: *Von Punktwolken zu Dreiecksnetzen.* M.S. thesis, Universität Karlsruhe, Germany, 2002.
42. Chris Prince: *Progressive Meshes for Large Models of Arbitrary Topology.* M.S. thesis, University of Washington, Seattle, 2000.
43. S. Röttger, W. Heidrich, P. Slusallek, H.-P. Seidel: *Real-Time Generation of Continuous Levels of Detail for Height Fields.* Proceedings of 6th International Conference in Central Europe on Computer Graphics and Visualization '98, 315–322, 1998.
44. Szymon Rusinkiewicz, Marc Levoy: *QSplat: A Multiresolution Point Rendering System for Large Meshes.* Proceedings of SIGGRAPH '00, 2000.
45. Eric J. Stollnitz, Tony D. DeRose, David H. Salesin: *Wavelets for Computer Graphics: Theory and Applications.* The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling, Brian A. Barsky, Series Editor, 1996.
46. Julie C. Xia, Amitabh Varshney: *Dynamic View-Dependent Simplification for Polygonal Models.* Proceedings of the IEEE Visualization '96, 327–334, 1996.
47. Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, Markus Gross: *Surface Splatting.* Proceedings of SIGGRAPH '01, 2001.